IN THE CLAIMS

Please amend the claims as follows:

1. (Currently Amended) An application programming interface for a programmable graphics processor pipeline, comprising:

one or more program instructions to configure a fragment processor within the programmable graphics processor pipeline to detect a position conflict for a an x,y position and prevent a subsequent access of the position until the position conflict is resolved, the instructions further including one or more instructions to write the x, y position and read the x, y position without an intervening instruction to flush the graphics pipeline.

2. (Original) The application programming interface of claim 1, wherein a program instruction receives as input a source location and a destination location.

3. (Original) The application programming interface of claim 2, wherein the source location includes a buffer identifier corresponding to one of several buffers.

4. (Original) The application programming interface of claim 2, wherein the destination location includes a buffer identifier corresponding to one of several buffers.

5. (Original) The application programming interface of claim 2, wherein the destination location contains fragment data including at least one of depth, color, and stencil.

6.-10. (Cancelled)

11. (Original) A fragment program for processing fragment data in a fragment processing pipeline, comprising:

a fragment program instruction to write a destination location in a buffer; and

a fragment program instruction to read the destination location in the buffer, without an intervening instruction to flush the fragment processing pipeline.

12. (Original) The fragment program of claim 11, wherein the destination location includes a buffer identifier corresponding to one of several buffers.

13. (Original) The fragment program of claim 11 comprising fragment program instructions to configure the fragment processing pipeline to perform depth buffering prior to shading.

14. (Original) The fragment program of claim 11, comprising fragment program instructions to configure the fragment processing pipeline to perform depth peeling.

15. (Original) The fragment program of claim 11, comprising:

fragment program instructions to configure the fragment processing pipeline to perform raster operations.

16. (Currently Amended) The fragment program of claim 11, wherein [[the]] raster operations are performed using fragment data represented in a floating-point data format.

17.-22. (Cancelled)

23. (New) A method for processing fragments in a graphics processor pipeline, comprising:

providing a fragment processing unit within the graphics processor pipeline;

receiving a first fragment associated with a position by the fragment processing unit;

processing the first fragment associated with the position to obtain a processed first fragment;

receiving a second fragment associated with the position by the fragment processing unit;

interlocking the second fragment in part subject to completion of the processing of the first fragment;

writing the processed first fragment to a graphics memory;

unlocking the second fragment; and

processing the second fragment in the fragment processing unit without flushing the pipeline between processing the first and second segments.

24. (New) A method as claimed in claim 23, including processing one or more additional fragments following processing the first fragment without unlocking the second fragment.

25. (New) A method as claimed in claim 23, including specifying the position of the first segment as source data for subsequent processing of fragments.

26. (New) A method as claimed in claim 25, wherein the interlocking step comprises reaching the source data prior to processing the second fragment to prevent writing to the position.

27. (New) The application programming interface of claim 1, wherein the position comprises a region including a plurality of pixels.

28. (New) A method as claimed in claim 23, including checking a location in graphics memory for the processed first fragment prior to unlocking the second fragment.

29. (New) A method as claimed in claim 23, including processing the first and either the second or the additional fragments in parallel.

30. (New) A programmable graphics processor for execution of program instructions, comprising:

a read interface configured to read data from a graphics memory;

a fragment processing unit configured to receive fragments, each fragment associated with a position, and the data from the graphics memory and generate processed fragments;

a conflict detection unit configured to selectively store the position associated with each fragment and generate a position conflict status;

a write interface configured to write the processed fragments to the graphics memory; and

a fragment processing pipeline configured to handle read-after-write hazards during execution of shader programs including an instruction to write a location in graphics memory, an instruction to check the location in a conflict detection unit and a subsequent instruction to read a location in graphics memory without an intervening instruction to flush the fragment processing pipeline based on the check of the conflict detection unit.

31. (New) A programmable graphics processor as claimed in claim 30, including a data cache storing additional data associated with the location, the conflict detection unit determining if the additional data associated with the location is available.

32. (New) A programmable graphics processor as claimed in claim 31, wherein the location is a region comprising a plurality of pixels.